Rapport labo01

3 Interlacement sur std : :cout

- Après insertion de la ligne `std::cout << "Thread number : " << tid << endl;` on remarque que le numéro de thread actuel est affiché continuellement. Cependant le changement de thread s'effectue parfois en plein milieu du print de la ligne.
- Après insertion de la ligne `logger() << "Thread number : " << tid << endl;` chaque print à le temps de se terminer avant le changement de thread.
- Cela est du au fait d'avoir un lock_guard au début du destructeur de l'objet logger. Lorsque l'on utilise logger pour print dans le shell le lock_guard garantit l'éxecution complète du cout et détruit l'objet.

4 Ordre d'exécution

- Je ne pense pas qu'il soit possible de prédire la valeur de fin dans l'état actuel du code.
- Le résultat est soit 1 soit 0 mais jamais 2.
- Comme le résultat est aléatoire la reproductiblité de ce programme est douteuse.
- Ajout de :

```
while(inOrder != tid);
sharedVar = tid;
++inOrder;
```

Dans la fonction run.

5 Timestamps

- Lorsqu'il n'ya q'un seul coeur, le cpu execute les thread en alternant des un aux autres. Avec deux coeur le travail est partager et deux thread peuvent être traiter en meme temps.

6 Incrémentation de compteur

- Afin de résoudre le problème nous avons passer la variable counter dans le type `std::atomic`. Selon la documentation ce type permet la synchronisation inter-thread lors de l'accès un objet de ce type. Nous obtenons ainsi un ratio de 100% pour les test suivant: 1 coeur:

Nbr de thread	Nbr d'iteration	Ratio
1	1	100%
100	1	100%
10000	1	100%
10000	100	100%
10000	10000	100%

2 coeurs:

Nbr de thread	Nbr d'iteration	Ratio
1	1	100%

Nbr de thread	Nbr d'iteration	Ratio
100	1	100%
10000	1	100%
10000	100	100%
10000	10000	100%